electric monk: a system for auto-deriving spatial moment equations

Ben Bolker

January 22, 2009

1 Introduction

electric monk (em for short: named after the character in a Douglas Adams novel)[1] is a collection of Mathematica code for automatically deriving, analyzing, and numerically integrating spatial moment equations. em's core consists of five sets of rules:

- rules for finding the expected changes of variables and products of variables, given a set of transitions and transition rates;
- rules for simplifying integrals involving *kernels* (probability distributions of interaction strengths as a function of distance), means, covariances, etc.;
- rules for making *moment closure* approximations of these equations;
- rules for converting the resulting moment equations from Mathematica to input files for `xtc`, a numerical integration program by Bard Ermentrout;
- rules for simplifying combinations of Fourier-transformed Laplacian/Bessel kernels.

This package is intended to make the process of exploring spatial moment equations less painful. As with any free package, it comes with NO WARRANTY, expressed or implied. (It is released under the Gnu Public License, in case that means anything to you.) As with any software package for doing "analysis", it is your responsibility not to do anything silly, and to understand what your results mean ...

[1] "The Electric Monk was a labour-saving device, like a dishwasher or a video recorder. Dishwashers washed tedious dishes for you, thus saving you the bother of washing them yourself, video recorders watched tedious television for you, thus saving you the bother of looking at it yourself; Electric Monks believed things for you, thus saving you what was becoming an increasingly onerous task, that of believing all the things the world expected you to believe": Douglas Adams, *Dirk Gently's Holistic Detective Agency*

2 Starting up

Start Mathematica (either in text or notebook mode) and load in the package:

```
<<emonk.ma
```
(1)

To get help on a particular function `foo` (provided I've written it) type `?foo`; for example,

```
?meaneq
```
(2)

3 Defining a model

Enter the transition rates for an individual-based model as a nested list in Mathematica form as follows:

```
model = {rule1,rule2,rule3}
```
(3)

where each of the rules specifies a transition rule as defined below. (Note that each rule is itself a list in Mathematica, so it will have its own set of curly brackets {} around it.)

Each transition rule is defined as a list of three components. The first component is the list of variables (states) and locations changed by the transition; the second is the (list of) sizes of the changes to each variable made by the transition (typically $\pm$ 1); and the third is the rate of the transition (a formula, *not* a list) expressed as a function of local and nearby state values.

3.1 Local (density-independent) removal/addition

For example, a simple mortality term

$$N(x) \to N(x) - 1 \text{ at rate } \mu N(x) \tag{4}$$

would be expressed in the Mathematica model as

```
rule1={{N[x]},{-1},{mu*N[x]}}.
```
(5)

3.2 Local (density-independent) transformation

A density-independent transformation such as germination (from fruit (`N`) to seedling (`S`)) would be expressed mathematically as

$$\{N(x), S(x)\} \to \{N(x) - 1, S(x) + 1\} \text{ at rate } gN(x) \tag{6}$$

and in Mathematica as

```
rule2={{N[x],S[x]},{-1,+1},g*N[x]}.
```
(7)

3.3 Neighbourhood- or (density-)dependent addition or removal

Transitions can also depend on neighborhood densities, expressed as integrals over kernels. Fruit deposition from neighboring trees (`T`) which have (passive) dispersal kernel D and seed production rate s would lead to a transition

$$N(x) \to N(x) + 1 \text{ at rate } \int_\Omega sD(|y - x|)T(y)\,dy \tag{8}$$

(where Ω is the entire space, x and y are vectors, and $|x-y|$ is the distance between them) or

```
rule3={{N[x]},{+1},int[s*Dk[y-x]*T[y],y]}
```
(9)

(the integration variable, y in this case, is given as the second argument to the integral function `int`). (Note that the kernel is also expressed as a function (using square brackets in Mathematica); we use `Dk` instead of `D` because the latter is a reserved word in Mathematica.)

3.4 Neighbourhood- (or density-)dependent transformation

Infection of a susceptible individual (`N`) by propagules from infected individuals (`I`) in the neighborhood is a transformation that depends on local neighborhood density of infectives

$$N(x) \to N(x)-1, I(x) \to I(x)+1, \text{ at rate } \beta N(x) \int_\Omega U(|y-x|)I(y)\,dy \quad (10)$$

or

```
rule4={{N[x],I[x]},{-1,+1},beta*N[x]*(int[U[y-x]*I[y],y])}
```
(11)

3.5 Movement

Movement has to be specified as a pair of transitions, the local emigration of individuals (which in the simplest case depends only on local density, but which could be dependent on neighbourhood density of the same or other species) and the neighbourhood-dependent immigration of the same individuals. Technically this preserves only statistical balance rather than detailed balance of the numbers of individuals, but since we are deriving equations for the statistics of the population we can get away with fudging this. (Another slight fudge is necessary; you have to tell the package explicitly which kernels represent movement (see §6.1), to avoid having it think that the immigration rate is a reproduction term which would require a singular term in the covariance equations.)

Mathematically:

$$N(x) \to N(x)-1 \text{ at rate } -mN(x) \quad (12)$$

and

$$N(y) \to N(y)+1 \text{ at rate } m \int_\Omega D_m(|y-x|)N(x) \quad (13)$$

or in Mathematica:

```
rule5={{{N[x]},{-1},-m*N[x]},{{N[y]},{+1},m*int[Dm[y-x]*N[x],x]}}
MoveKernList={Dm}
```

3.6 Exogenous heterogeneity

In order to put (static) exogenous heterogeneity into the model, one just has to define the exogenous state variable as a static variable, and then proceed to use it in the definitions of other state transitions. The only trick is that one has to let `em` know that there is no additional Bernoulli/Poisson variance at small scales (so far this small-scale variance is required to be zero, this may change in the future).

For example, a model with density-independent but spatially heterogeneous mortality rates:

$$\begin{aligned} \frac{d\mu(x)}{dt} &= 0 \\ \lim_{r\to 0}\int_0^r c_{\mu\mu}(r')\,dr' &= 0 \\ N(x) &\to N(x)-1 \text{ at rate } \mu(x)N(x) \end{aligned} \tag{14}$$

or

```
rule6={{{mu[x]},{0}, 0},{{n[x]},{-1}, mu[x]*n[x]}}
ContVarList={mu}
```

A straightforward extension in this case is to have organisms sense the environment, not just at a particular point, but in a neighborhood around themselves:

$$N(x) \to N(x)-1 \text{ at rate } N(x)\int H(|y-x|)\mu(y)\,dy,$$

or

```
|{{n[x]},{-1}, n[x]*int[H[y-x]*mu[y],y]}|
```

(this model can also be represented by simply convolving the habitat heterogeneity with the habitat kernel H to begin with and then treating the result as a static distribution that is sensed at a particular point ...)

I've never built a model with dynamic heterogeneity, but the way to go about it would be to define a dynamic process that ended up with the appropriate mean and covariance at equilibrium, then to incorporate it as a dynamic (rather than a static) state variable. (This is essentially what you get with completely asymmetric competition; to the weaker competitor, the static competitor represents a form of exogenous spatiotemporal variation.)

4 Technical details

4.1 Variable names

My conventions are to specify state variables as single uppercase letters; kernels as an uppercase letter combined with lowercase letters or numbers (`Dk` for dispersal, `Mk` for movement, `Uc`, `Ui`, `U11` etc. for different competition/infection interactions); spatial variables as `w`, `x`, `y`, `z`, etc.. I have spelled out Greek letters rather than using Mathematica's fancy built-in system for special characters just because I'm old fashioned.

Please keep the following restrictions in mind when picking variable names — it's easy to get yourself in trouble here ...:

1. Variables must be legal Mathematica symbols (e.g. punctuation within names is not allowed);

2. Avoid Mathematica reserved words (`D`, `E`, `I`, `N`, and `O` are all functions or mathematical constants in Mathematica);

3. Avoid other reserved words that I have used in this package: `int`, `conv`, `M`, and `c` should probably be avoided ... (others??)

4. **Position variables**: unfortunately, there are fairly strict conventions on position variables. I usually take `x` to be the focal site, `y` to be a site that interacts with the focal site (via competition, dispersal, etc.), and `z` to be the location for which covariances are calculated: `v` and `w` are auxiliary interaction locations (for instance in the case of density-dependent fecundity where a parent at `y` dispersing its offspring to `x` may experience competition with site `w` [!]) These conventions are mostly hard-coded, although I have started to try to relax them. For example, `x` is automatically set to zero when taking integrals, to incorporate the effect of spatial homogeneity; expectations must be taken with the same position variable as is used in the state transition definition, or the expected rates will come out zero (e.g. in the `posvars` argument to `meaneq` or the `covvars` argument to `coveq`).

5. If you plan to translate your equations to `xtc` format (see below), note that `xtc` is **not case-sensitive** (so `f` and `F` will get confused), and may have some of its own reserved words (`exp`, `gamma`, etc.).

4.2 Miscellaneous

`TableForm[model]` will give a prettier display of the model definitions. (There are some sketchy functions to translate state variables etc. to LaTeX format.)

5 Example so far

Now that you've gotten this far, here's a very simple example: the rules for the "spatial logistic equation" with density-dependent mortality. In this model, we assume that individuals experience density-independent mortality at *per capita* rate μ (i.e., the probability of an individual dying in a small interval Δt is $\mu \Delta t$); they experience an additional density-dependent mortality, linearly increasing with slope α as a function of local density weighted by a competition kernel U; and they reproduce at constant *per capita* rate f, after which their offspring (seeds, whatever) immediately disperse away from the parent with dispersal kernel D.

In mathematical form:

transition	**rate**	**description**
$N(x) \to N(x) - 1$	$\mu N(x)$	density-independent mortality
$N(x) \to N(x) - 1$	$\alpha N(x) \int U(y-x) N(y)\, dy$	density-dependent mortality
$N(x) \to N(x) + 1$	$f \int D(y-x) N(y)\, dy$	reproduction and local dispersal

Enter the model in Mathematica like this:

```
spatlogist = {{{N1[x]},{-1},mu*N1[x]},
              {{N1[x]},{-1},N1[x]*alpha*int[U[y-x]*N1[y],y]},
              {{N1[x]},{+1},f*int[D1[y-x]*N1[y],y]}}
```

And here's what a little piece of the Mathematica session looks like:

```
In[5]:= In[5]:= TableForm[spatlogist]

Out[5]//TableForm= N1[x]   -1   mu N1[x]

                   N1[x]   -1   alpha int[N1[y] U[-x + y], y] N1[x]

                   N1[x]   1    f int[D1[-x + y] N1[y], y]
```

A few notes:

- you can combine density-dependent and density-independent mortality in a single rule, but it seems clearer if you separate them;
- avoid the temptation to use `N` as a state variable or `D` as a dispersal kernel — they're both reserved words in Mathematica.
- Mathematica sorts variables in expressions alphabetically (so, for example, the kernel `D1` comes before the state variable `N1` in the integral, but `U` comes after).

6 Generating moment equations

6.1 Variable and kernel lists

The first step is to extract lists of state variables, spatial position variables, and kernels from the equations (*this should now be done completely automatically*). State variables are extracted from the first entries in each transition definition; kernels are identied as any Mathematica function in the transition rates that is neither an integral nor a state variable.) If you should need to do this manually for some reason,

```
VarList=getvarlist[model],          (15)
```

```
KernList=getkernlist[model],        (16)
```

and

```
XVarList=getxvarlist[model]         (17)
```

will extract the variables, kernels, and position variables respectively.

In order to do the bookkeeping correctly, movement kernels do need to be specified separately by setting e.g. `MoveKernList={Mk}`, and continuous (non-discrete) environmental variables need to be specified as part of `ContVarList`.

6.2 Mean and covariance equations

Once you've defined your model you can generate the moment equations simply by saying

```
meaneqs[model]
```
(18)

to get the equations for time-evolution of the mean densities (RHS of ordinary differential equations for each state variable's density in order of `VarList`, which is alphabetical). Mean densities of state variables are denoted by `M[n]`. The expression `cbar[U,m,n]` denotes an kernel-weighted average covariance, corresponding to $\int U(r)c_{mn}(r)\,dr$. "Naked" covariances c_{mn} should not appear in the mean equations!

Use

```
coveqs[model]
```
(19)

for the covariance equations (RHS of integral equations in pairwise order). The spatial lag variable r is implicit throughout these equations. Covariances of state variables are `c[m,n]`; convolutions between covariances and kernels are `conv[c[m,n], K, z]` $(= (K * c)(r) = \int U(s)c_{mn}(r+s)\,ds)$. (FIXME: The spatial variable `z` is redundant/implicit and should be dropped. `implicitvar`?) Third central moments (which always appear inside an integral, `int`) `M3` depend on three state variables.

If you want to see exactly what Mathematica is doing to reduce your model to moment equations, try setting `exdebug=True` and running `meaneqs` or `coveqs` again ... `intdebug=True` should provide information on the integration rules.

6.3 Moment closure

closeM3 is now closeM!

Third central moments `M3` have to be dealt with before you can proceed to analyze or numerically integrate the equations. The function for performing moment closure on a set of equations is `closeM3`. It has a series of options that can be set; the most important is `closure`.

- To set third central moments to zero (power-1 closure), use `closure->power1`.
- To use an asymmetric power-2 closure (where we decompose the conditional probability of a triple XYZ into $p_{XYZ} = p_{XY}p_{YZ}/p_Y$), use `closure->power2a`. There is an additional option for `power2a`, `closepos` (default `0`), which specifies the spatial location of the "middle" of the triple.
- symmetric power-2 closure uses

$$p_{XYZ} = \frac{1}{2}\left(\frac{p_{XY}p_{YZ}}{p_Y} + \frac{p_{XZ}p_{YZ}}{p_Z} + \frac{p_{XY}p_{XZ}}{p_X} - p_X p_Y p_Z\right).$$

  The last term inside the parentheses is necessary for consistency (REF Dieckmann and Law, IIASA book).

- weighted power-2 closure is a generalization of the preceding two closure, due to Murrell, Law and Dieckmann:

$$p_{XYZ} = \frac{1}{\alpha+\gamma}\left(\alpha\frac{p_{XY}p_{YZ}}{p_Y} + \beta\frac{p_{XZ}p_{YZ}}{p_Z} + \gamma\frac{p_{XY}p_{XZ}}{p_X} - \beta p_X p_Y p_Z\right).$$

  There is an extra parameter, `closeweights`, which defaults to $\{\alpha, \beta, \gamma\} = (4, 1, 1)$: `closeweights`=(1,0,0) corresponds to `power2a` and `closeweights`=(1,1,1) corresponds to `power2s`.

- To do power-3 closure, making the assumption that $p_{XYZ} = \frac{p_{XY}p_{YZ}p_{ZX}}{p_X p_Y p_Z}$: use `closure->power3`.

`closeM3` also handles what I call "third-moment singular terms", which arise when the triple XYZ is really just a pair ($X = Z$, etc.) or a single location ($X = Y = Z$). These third-moment singular terms are taken into account if `singm->True`, which is the default; set `singm->False` to ignore them. (Since a typical third-moment singular term S appears in the form $K(r)S(r)$ rather than the form $\int K(r)S(r)\,dr$, one can argue that third-moment singular terms are of the same order as the scale parameter and can be neglected when all kernels are relatively long-range: Bolker and Pacala 1999 make this assumption.)

Singular terms can only arise between a pair $X(x)Y(y)$ in a triplet $X(x)Y(y)Z(z)$ if (1) the state variables X and Y are identical (otherwise the locations x and y could not refer to the same point, because it would have to be occupied by two different individuals) and (2) x and y are not the lag variables involved in the current covariance calculation (otherwise this singular term would only appear in the covariance at lag zero, which we do not calculate anyway). The option `covpos` (default `{0,z}`) specifies the lag variables for the current covariance calculation, so that criterion 2 can be determined.

It is often useful to simplify the equations further after applying moment closure rules: something like

```
simpeqs = eqs //. intrules
```

will apply a set of rules for simplifying integrals. There is an important "state variable" called `singtaken`; if `True`, it tells Mathematica that singular terms have already been taken out of all covariances and third moments (so it doesn't try to extract them again). `closeM3` has a `setsingtaken` option (`True` by default) that sets `singtaken=True`. Set `singtaken` back to `False` if you want to derive more moment equations from transition rules in the same Mathematica session. **perhaps `meaneqs`/`coveqs` should automatically toggle `singtaken` back to `False`?**

At this point you may also want to use the function `implicitvar`, which drops a lag variable (by default `z`) from the equations and makes it implicit. For example, `c[s,s,z]` (which may already have been implicitly shortened from `c[s,s,0,z]`) gets shortened to `c[s,s]`. Once you reach this stage there should be no explicit spatial position variables left in the equation.

6.4 Scaled-covariance and correlation equations

The equations produced at this point are equations for the covariance density. You may prefer to work with the equations in another scaling, particularly if you plan to analyze invasion criteria.

Here are some of the possible scalings one could use (not all of which are available in this package):

probability	p_{xy}
conditional probability	$p_{x\|y} = \frac{p_{xy}}{p_y}$
covariance	$C_{xy} = p_{xy} - p_x p_y$
scaled covariance	$SC_{xy} = \frac{C_{xy}}{p_x} = \frac{p_{xy}-p_x p_y}{p_x} = p_{y\|x} - p_y$
correlation	$c_{xy} = \frac{C_{xy}}{p_x p_y} = \frac{p_{xy}-p_x p_y}{p_x p_y} = \frac{p_{xy}}{p_x p_y} - 1$
"multiplicative covariance" (?)	$\frac{p_{xy}}{p_x p_y} = c_{xy} + 1$

8 June 2002: changed order/definition of scaled covariance so that SC_{xy} is now proportional to neighborhood density/conditional probability of y near x, for consistency with competition coefficients, so that $\alpha_{xy}\overline{SC_{xy}}$ is the competitive effect of y on x ...

```
covtocor[coveqs,Teqs]
```

translates a set of covariance equations `coveqs` derived from an original set of transition probabilities `Teqs` to correlation equations. (You need to specify the original transition equations again because the translation involves the expressions for changes in the mean densities as well as covariances.)

Similarly,

```
covtoscov[coveqs,Teqs]
```

translates to scaled covariances. Note that there will be more scaled-covariance equations than covariance equations (n^2 rather than $n(n+1)$), because scaled covariances are not symmetric ($SC_{xy} = p_x/p_y SC_{yx}$). You will also need something like

```
meqs /. covcorrules
```

to translate any covariance (or averaged covariance) terms in the mean equations to scaled equivalents.

7 Analyzing models

You can take your equations at this point and do whatever you like with them: analyze equilibria, quasi-equilibria, integrate the equations numerically, etc.. If you choose to continue the analysis within Mathematica, there are various Mathematica commands (built-in or included in the `em`) that may come in handy. These are just a few hints: for complete details on analysis you'll have to read the various Mathematica documentation.

- Use the pattern rule `/.` to substituting simplifying combinations of parameters or numerical values of parameters. To apply multiple rules you may want to use `Flatten`. Examples:

  ```
  simpeqs = Simplify[eqs /. {h->(rho-d-r), alpha->A-m}];
  simpeqs = eqs /. {a -> 1.3, b-> 2.7}
  ```

- `fourtrans`: apply a symbolic Fourier transform to covariance equations (all this function does is to transform convolutions into products and *vice versa*; it's only really worth it if the only covariance terms in your equations are in terms of $\bar{c}$ terms or convolutions, in which case you will only have products in the Fourier-transformed equations).
- `zerocbar`: drop spatial terms in mean equations, leading to the mean field model
- `sameK`: set all kernels equal to each other
- Substitute functional forms of kernels (in space or frequency domain), e.g. `/. K -> 1/(1+q^2)`
- substitute mean-field solutions in covariance equations (approximate solution when effects of space are small → densities are close to MF values)
- `momcollect`: collect covariance (and possibly third-moment) terms in a set of equations.

7.1 Equilibrium/quasiequilibrium solutions

Integration of `cbar`, etc..

7.2 Kernel identities

If we assume Laplacian kernels ($K(r) = \lambda/2e^{-\lambda|r|}$) in one dimension or modified-Bessel kernels ($K(r) = K_0(\lambda r)$) in two dimensions, we can get some mileage out of a set of *kernel identities*. What is special about these kernels is that they have the same convenient functional form in the frequency domain, $\lambda^2/(\lambda^2+q^2)$. "Rational functions" of kernels (e.g., of the form $K_1/(c+K_2)$) in the frequency domain can be reduced by partial fractions to somewhat simpler forms, and I've encoded many of these rules in Mathematica(see `kern1.ma` [which probably needs significant cleanup and documentation before anyone else could bear to look at it]).

7.3 Translation to `xtc`

The package can output any equations you have derived as an input file for `xtc`, a program by Bard Ermentrout that does numerical integration of one-dimensional integral equations (URL `http://www.math.pitt.edu/~bard/bardware/xtc-1.3.tar.Z`). Many of the combinations of moment closure/Fourier transform, along with suitable assumptions about radial symmetry, allow the moment equations to be written as 1D integral equations. Of course, any equations for an explicitly 1-D system can be written as 1-D equations. In addition, the Fourier transformed version of MEs with power-1 closure or the regular (space domain) version of MEs with power-2A closure come out as 1-D equations, with the added bonus that the equations contain no explicit convolutions, which are possible, but slow, in `xtc` (although you may have to drop third-moment singular terms to make this work).

I had written a whole set of C codes for numerically integrating moment equations, but I have turned to `xtc` because it is more flexible. I have fixed some bugs in `xtc` and added some functionality (batch mode, ability to read in starting conditions for continua from a file); for the time being, you can get my version (latest version is 1.3c) from my web page. In order to run `xtc` on Windows, you will have to compile it (it comes as fairly portable C code) and get yourself an X-windows emulator to run the graphics (see ...). I intend to write a version of `xtc` that can be compiled with graphics turned off (and run only in batch mode), but this hasn't happened yet. Ermentrout's `xppaut` package, for numerical analysis of ODEs, is far slicker and has had a lot of work in the past few years. He's adapted it to solve most of the PDE-like problems he is interested in, so upgrades for `xtc` (which is written and distributed entirely at his own initiative and expense) are less likely to happen. Nevertheless, I find `xtc` sufficiently powerful and convenient that I prefer it to `xppaut` (or to my own code) for these applications.

The command `mmatoxtc[meqs,coreqs,parvals,cparvals,mstart,opts,xtcopts,backup]` produces an `xtc` input file with a given set of mean equations, covariance/scaled covariance/correlation equations, parameter values, continuum parameters, starting values for the mean densities, and options. Parameters are expressed in the form `{p1->1.0,p2->2.0}`, etc.; kernel parameters have to be expressed as `KP1` for the first parameter for kernel `K`, etc.. Remember that `xtc` is not case sensitive, so choose your variable names with care. Options are expressed as `option->value`: your choices are

- dimension, `dim` (1 or 2);
- Fourier-transformed equations, `ftrans` (`True` or `False`);
- whether the equations are for scaled covariances rather than covariances (in which case c_{12} and c_{21} are not identical): `sccov=False` by default
- output file, `outfn` (a quoted file name: note that the code automatically makes numbered backups if you try to overwrite an existing file);
- total arena length (`length=10` by default) and grid scale (`griddx`=0.02 by default);
- kernel shapes: defaults are
  - `cauchy`=$\lambda^2/(\lambda^2+q^2)$ if Fourier-transformed
  - `mexp`=$\lambda/2\exp(-\lambda x)$ (back-to-back exponential, renamed to avoid conflicting with `xtc`'s own exponential function) in 1 dimension
  - or `mexp2d`=$\lambda^2/(2\pi)\exp(-\lambda r)$ in 2 dimensions.

  The other choices are
  - `gauss`(σ)=$1/(\sqrt{2\pi}\sigma)\exp(-(x/\sigma)^2/2)$
  - `gauss2d`(σ)=$1/(2\pi\sigma^2)\exp(-(r/\sigma)^2/2)$
  - `beta`(a,b)=$H(1-x)\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1}$

  - `sbeta`(a,b,s) [scaled beta]=$H(s-x)\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}(x/s)^{a-1}(1-(x/s))^{b-1}$
  - `errdist`(τ,d)=$\frac{\Gamma(2/\tau)}{2\tau d(\Gamma(1+1/\tau))^2}\exp(-(x\Gamma(2/\tau)/(d\tau\Gamma(1+1/\tau)))^\tau)$ (a "generalized Gaussian" distribution with mean dispersal distance d and exponent τ: proportional to exponential with $\tau = 1$, Gaussian with $\tau = 2$)

- starting values of covariances, `cstart`
- various `xtc` settings (as a list of `xtcopts`): `deltat`, `tfinal`, `nout`, `method`, `fast`, `bound`, `bufsize`, `transient`; you can also set these within `xtc` (see `xtc` documentation for details).

For example:

```
mmatoxtc[meqs,ceqs2,{f->5,mu->1,alpha->1,UP1->1,DkP1->1},{},{0.1},0,
{dim->1,ftrans->False,outfn->"DDM1a.xtc"},{nout->500}]
```

You will have to read the documentation of `xtc` for details of how to run and analyze the numerical models, but the basic steps are:

- `xtc foo.xtc` at a shell prompt;
- **G** for Go, to run the equations;
- **W**indow/**F**it 3d to adjust colors in the plot;
- **2**D graphics/**4** (U vs. t) to bring up a dialog box for displaying scalar quantities (densities or $\bar{c}$ values), and **W**indow/f**I**t 2d to adjust/display the plot;
- **R**edraw menu to redraw 2- or 3d plots after they are changed or obscured by other windows;
- **F**ile/**S**ave 3d to save data shown in the 3d plot, or **F**ile/s**A**ve 2d for the 2d plot;
- **F**ile/**Q**uit/**Y**es to stop.

You can also call sets of batch runs from `xtc`.

- `runxtc` is the basic function for running moment equations in `xtc` from Mathematica. It writes out an output file, calls the operating system to run `xtc`, and returns the results (*describe format*).
- `findxtceq0` runs equations to equilibrium, returning the last two time steps
- `findxtceq` runs equations to equilibrium, checks for convergence, and returns the state at the last time step
- `findxtcinv` runs an invasion analysis in `xtc`, running one species to monoculture equilibrium and then introducing the other at low densities. **Parameters:**

  - mean equations
  - covariance equations
  - parameter values
  - starting values for mean (list of same length as mean equations)
  - starting values for covariances

- `findxtceqtab` finds the equilibrium for a series of parameter values (specify the parameters as a list of sets of parameters)

- `findxtceqstab` finds the equilibrium for a series of starting values

The order of variables in the output is: (time, mean 1, ..., mean n, { cov 1 }, ..., { cov n }, cbar 1, ..., cbar n). Note that `findxtceqtab` goes to some trouble to start each new run from the *ending* condition of the previous run, so that equilibrium should be achieved faster (if changes in parameters from run to run are fairly gradual).

Some other notes about running moment equations under `xtc`:

- the Fourier-mode and 2D calculations are not tested yet

- length is set in two places in the `xtc` file (as an option and as a parameter), be careful to change them both if you are editing the file

- You do have to be careful when numerically integrating moment equations:
  - power-1 closures will often blow up if you start the mean too small (this is the fault of the closure itself, not the numerical methods)
  - if you don't set the length of the arena large enough, you may have problems because of the way `xtc` extrapolates beyond the edge of the arena. If the value of the covariance at the maximum distance is not close to zero, watch out

- you can't use an expression within a convolution — this is OK except for multiple convolutions (e.g. $(Uc) * c$ or $(U * c) * c$), can be solved by adding another continuum fixed variable that defines the expression

- `xtc` wish list:
  - zoom/thinning capability for display of 3D surfaces
  - mixed extensions (e.g. even on left boundary, zero on right boundary)
  - label 2-D graph window with file
  - direct output (png?) of 3D color graph
  - more robust generally (e.g. segfaults if a scalar equation is undefined; crazy if one tries to use expressions within a convolution); better error reporting?
  - ability to define a default 2D graph

8 Wish list/notes

- what changes would actually be required for non-infinitesimal patches? Could you still assume Poisson statistics at a patch or would you have to make zero-dimensional (distributional) moment closure and track variance separately?
- need some rules for environmental variables with non-Poisson variances: this does come up occasionally (e.g. when organisms sense the environment through a "habitat kernel")
- directional dispersal??
- Note order of covariances etc.: alphabetical
- Further functions for quasi-eq/invasion analysis?? (some of these already exist)
- Fix/sort out sensitivity to spatial positions
- correlated dispersal??
- correlated disturbance??
- tests: power-1, power-2, power-3 closure, 1- and 2-D, with and without singular terms, space and frequency domain, DDM/DDE/DDF. Cf. simulations ... different parameter regimes ...
- dealing with self-competition: `U[0]->0`; works OK but is wrong in `xtc` output if we change to FT kernels and leave $U(0)$ unchanged
- FIXME: order of n1, n2 in Cbar variables is still getting switched around by mmatoxtc
- FIXME: change variables when taking Fourier Transform?
- FIXME: should `closeM3` automatically apply `intrules`?
- tests:
  - replicate equations/numerical solutions in Bolker & Pacala papers;
  - replicate equations/numerical solutions in Bolker '99;
  - replicate equations in Brown & Bolker ms.;
- how many possible models/numerical sims are there? 3 closures (power-1, power-2A, power-3) × singular terms (1,2) × dimension (1,2) × state variables (covariances, scaled cov., correlations) × number of species (1,2) × competition type (DDM, DDE, DDF) × special cases (neutral, symmetric, equal-scales, quasi-equilibrium, etc. etc.) ...

9 Examples

9.1 Disturbance

10 Tests

11 History

11.1 Changes to code:

11.2 Changes to documentation:

- July 2004: tweaking. Eliminated "tests" section (until I can get it a bit cleaned up)

12 Notes

6 July 2004:

- had some trouble with order of variables in an M3 term (not recognized) — had to use `resort`. Put in automatically at some point?
- must use `resort[c1,3] //.intrules` before trying to use `closeM`